Multicore Computer, GPU 및 Cluster 환경에서의 MATLAB Parallel Computing 기능

성 호현
MathWorks Korea
A Question to Consider

Do you want to speed up your algorithms?

If so…

– Do you have a multi-core or multi-processor computer?

– Do you have a high-end graphics processing unit (GPU)?
Agenda

Introduction to Parallel Computing Tools

- Using Multi-core/Multi-processor Machines
- Using Graphics Processing Units (GPUs)
Utilizing Additional Processing Power

- **Built-in multithreading**
  - In core MATLAB
  - For specific matrix operations
  - Automatically enabled since R2008a

- **Parallel Computing Tools**
  - Controlled by the MATLAB user
  - For a variety of applications
  - Leverage CPUs and GPUs to speed applications further
HKM Optimizes Just-in-Time Steel Manufacturing Schedule

Challenge
Optimize a steel production process to enable consistent, just-in-time delivery

Solution
Use MATLAB and Global Optimization Toolbox to maximize throughput of more than 5 million tonnes of steel annually

Results
- Algorithm development accelerated by a factor of 10
- Optimization time cut from 1 hour to 5 minutes
- Customer satisfaction increased

“C++, Java, or third-party optimization solutions would have required us to spend significantly more time in development or to simplify our constraints. Only MATLAB provided the flexibility, scalability, development speed, and level of optimization that we required.”

Alexey Nagaytsev
Hüttenwerke Krupp Mannesmann

Manually reviewed plant schedule (left) and plant schedule automatically optimized with MATLAB genetic algorithms (right). The optimized schedule minimizes schedule conflicts (in red), meets delivery dates, and achieves the target utilization rate.
Lund University Develops an Artificial Neural Network for Matching Heart Transplant Donors with Recipients

**Challenge**
Improve long-term survival rates for heart transplant recipients by identifying optimal recipient and donor matches

**Solution**
Use MathWorks tools to develop a predictive artificial neural network model and simulate thousands of risk-profile combinations on a 56-processor computing cluster

**Results**
- Prospective five-year survival rate raised by up to 10%
- Network training time reduced by more than two-thirds
- Simulation time cut from weeks to days

“I spend a lot of time in the clinic, and don’t have the time or the technical expertise to learn, configure, and maintain software. MATLAB makes it easy for physicians like me to get work done and produce meaningful results.”

Dr. Johan Nilsson
Skåne University Hospital
Lund University

Plots showing actual and predicted survival, best and worst donor-recipient match, best and worst simulated match (left); and survival rate by duration of ischemia and donor age (right).
Massachusetts Institute of Technology Integrates Cancer Research in the Lab and Classroom with MathWorks Tools

Challenge
Improve diagnostic techniques for cancer by identifying proteins and analyzing their interactions

Solution
Use MathWorks tools to enable students and researchers to analyze mass spectrometry data, model complex protein interactions, and visualize results

Results
- Education integrated with research
- Computation time shortened by an order of magnitude
- Research grant won

“Using a distributed approach with MATLAB code, we ran our analysis on a computer cluster and reduced computation time by an order of magnitude—from about a week to much less than a day.”

Dr. Gil Alterovitz
MIT and Harvard University

Link to user story
Research Engineers Advance Design of the International Linear Collider with MathWorks Tools

**Challenge**
Design a control system for ensuring the precise alignment of particle beams in the International Linear Collider

**Solution**
Use MATLAB, Simulink, Parallel Computing Toolbox, and Instrument Control Toolbox software to design, model, and simulate the accelerator and alignment control system

**Results**
- Simulation time reduced by an order of magnitude
- Development integrated
- Existing work leveraged

“Using Parallel Computing Toolbox, we simply deployed our simulation on a large group cluster. We saw a linear improvement in speed, and we could run 100 simulations at once. MathWorks tools have enabled us to accomplish work that was once impossible.”

Dr. Glen White
Queen Mary, University of London

Link to user story
Going Beyond Serial MATLAB Applications
Parallel Computing on the Desktop

- Use Parallel Computing Toolbox
- Speed up parallel applications on local computer
- Take full advantage of desktop power by using CPUs and GPUs
- Separate computer cluster not required
Scale Up to Clusters, Grids and Clouds

Desktop Computer

Parallel Computing Toolbox

Computer Cluster
MATLAB Distributed Computing Server
Scheduler
Parallel Computing enables you to ...
Agenda

- Introduction to Parallel Computing Tools
- Using Multi-core/Multi-processor Machines
- Using Graphics Processing Units (GPUs)
Programming Parallel Applications

Ease of Use

Greater Control
Using Additional Cores/Processors (CPUs)

- Support built into Toolboxes

Ease of Use

Greater Control
Example:
Built-in Support for Parallelism in Other Tools

- Use built-in support for Parallel Computing Toolbox in Optimization Toolbox
- Run optimization in parallel
- Use pool of MATLAB workers
Other Tools Providing Parallel Computing Support

- Optimization Toolbox
- Global Optimization Toolbox
- Statistics Toolbox
- Simulink Design Optimization
- Bioinformatics Toolbox
- Model-Based Calibration Toolbox
- …

*Directly leverage functions in Parallel Computing Toolbox*
Using Additional Cores/Processors (CPUs)

- Support built into Toolboxes
- Simple programming constructs: `parfor`
Running Independent Tasks or Iterations

- Ideal problem for parallel computing
- No dependencies or communications between tasks
- Examples include parameter sweeps and Monte Carlo simulations
Example: Parameter Sweep of ODEs

- Parameter sweep of ODE system
- Use pool of MATLAB workers
- Convert `for` to `parfor`
- Interleave serial and parallel code

Damped spring oscillator

\[ m \dddot{x} + b \ddot{x} + k \ x = 0 \]

\[ m, b, k \] 

- Sweep through different values of \( b \) and \( k \)
- Record peak value for each simulation
Using Additional Cores/Processors (CPUs)

- Support built into Toolboxes
- Simple programming constructs: `parfor`
- Full control of parallelization: jobs and tasks
Agenda

- Introduction to Parallel Computing Tools
- Using Multi-core/Multi-processor Machines
- Using Graphics Processing Units (GPUs)
Gaining Performance with More Hardware

Using More Cores (CPUs)

Using GPUs

Device Memory
What is a Graphics Processing Unit (GPU)

- Originally for graphics acceleration, now also used for scientific calculations
- Massively parallel array of integer and floating point processors
  - Typically hundreds of processors per card
  - GPU cores complement CPU cores
- Dedicated high-speed memory

* Parallel Computing Toolbox requires NVIDIA GPUs with Compute Capability 1.3 or greater, including NVIDIA Tesla 10-series and 20-series products. See [http://www.nvidia.com/object/cuda_gpus.html](http://www.nvidia.com/object/cuda_gpus.html) for a complete listing
Example:
GPU Computing in the Parallel Computing Toolbox

- Send and create data on the GPU
- Run calculations with built-in GPU functions
- Run a custom CUDA kernel
Benchmark: Solving 2D Wave Equation

CPU vs GPU

Graph showing the comparison between CPU and GPU for solving the 2D wave equation. The graphs are plotted on both linear and log scales, showing the time for 50 iterations versus grid size (N). The CPU performs significantly better than the GPU for smaller grid sizes, but the GPU becomes more efficient as the grid size increases.
Summary of Options for Targeting GPUs

- Use GPU array interface with MATLAB built-in functions
- Execute custom functions on elements of the GPU array
- Create kernels from existing CUDA code and PTX files
# Performance Acceleration Options in the Parallel Computing Toolbox

<table>
<thead>
<tr>
<th>Technology</th>
<th>Example</th>
<th>MATLAB Workers</th>
<th>Execution Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>matlabpool</td>
<td>parfor</td>
<td>Required</td>
<td>CPU cores</td>
</tr>
<tr>
<td>user-defined tasks (batch proc)</td>
<td>createJob, createTask</td>
<td>Required</td>
<td>CPU cores</td>
</tr>
<tr>
<td>GPU-based parallelism</td>
<td>GPUArray</td>
<td>No</td>
<td>NVIDIA GPU with Compute Capability 1.3 or greater</td>
</tr>
</tbody>
</table>
What else has changed?

**R2010b**
- Enhanced job manager security
  *(MATLAB Distributed Computing Server)*

**R2011a**
- Support for local workers
  *(MATLAB Compiler)*
- Integration with MATLAB desktop
  *(MATLAB, Parallel Computing Toolbox)*

**R2011b**
- Increase in the number of local workers to 12
  *(Parallel Computing Toolbox)*
- Support for GPU computation
  *(MATLAB Compiler)*
- Job Monitor
  *(Parallel Computing Toolbox)*
Summary

- Speed up parallel applications on desktop by using *Parallel Computing Toolbox*

- Take full advantage of CPU and GPU hardware

- If required, use *MATLAB Distributed Computing Server* to
  - Scale up to clusters, grids and clouds
  - Work with data that is too large to fit on to desktop computers
For more information

Visit

www.mathworks.com/products/parallel-computing